Planning for Hotfixes

Chris B. Behrens

SOFTWARE ARCHITECT

@chrisbbehrens

Introduction to Hotfixes

If we can rollback, we can fix things with less pressure

But that’s not always possible

Hotfix

a small piece of code developed to correct a major software bug or fault and released as quickly as possible.

1. **Small**
2. **Corrects a MAJOR bug**
3. **Released as quickly as possible**

A Major Fault

Jernathan Smith, CEO

- Faults which compromise the security of your application or enterprise
- Faults which make significant functionality unavailable to your customers
- Faults which may compromise the target system of the user

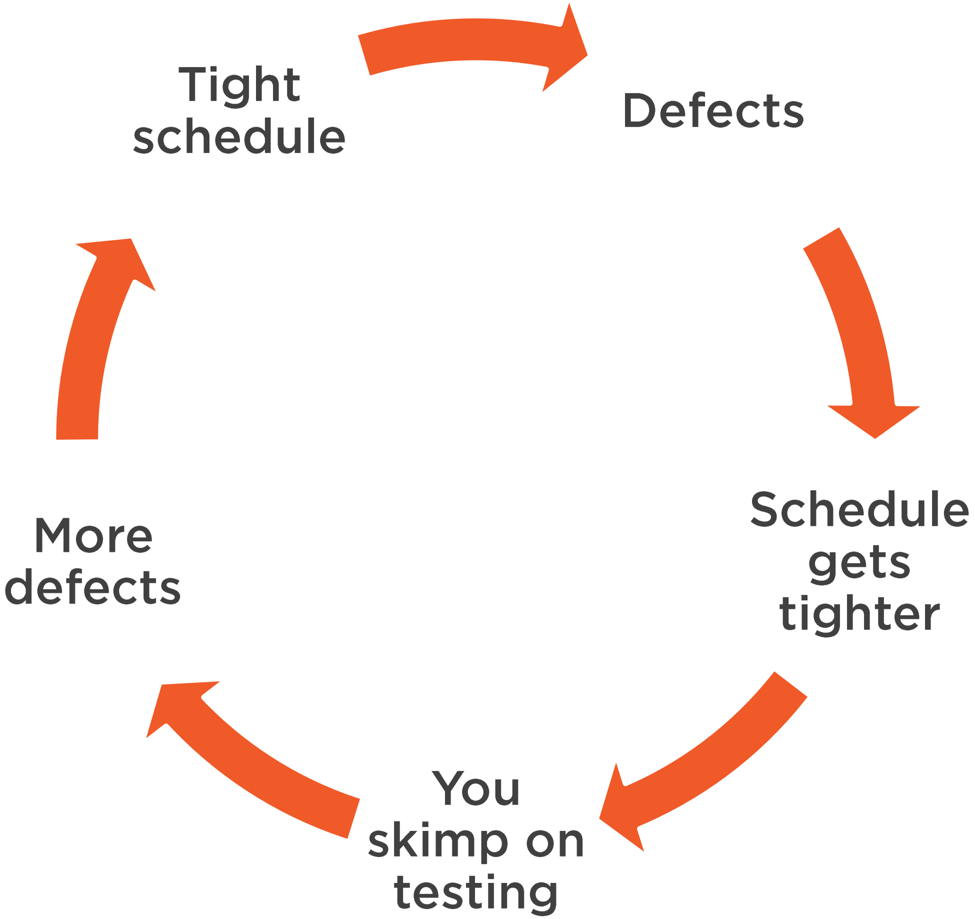
Everything should be made as simple as possible but not simpler.

- Not Albert Einstein

It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.

- Albert Einstein

The Hotfix Death Spiral



Tight schedule

Defects

Schedule gets tighter

You skimp on testing

More defects

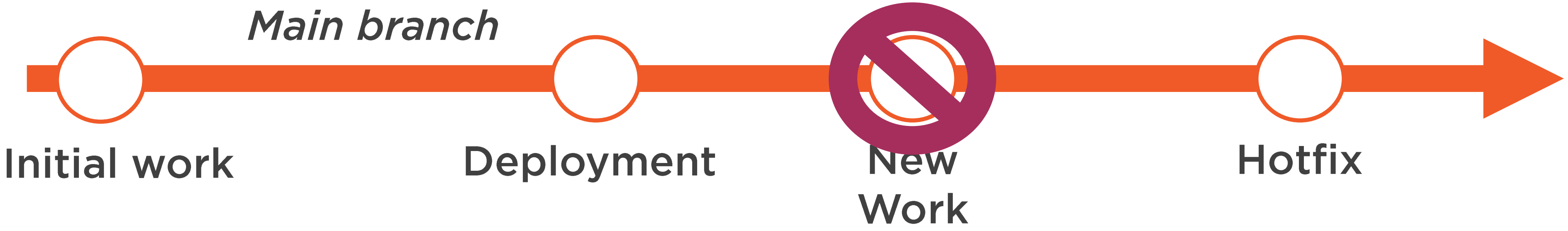
The Branching Model for Hotfixes

Hotfixing is not deployment

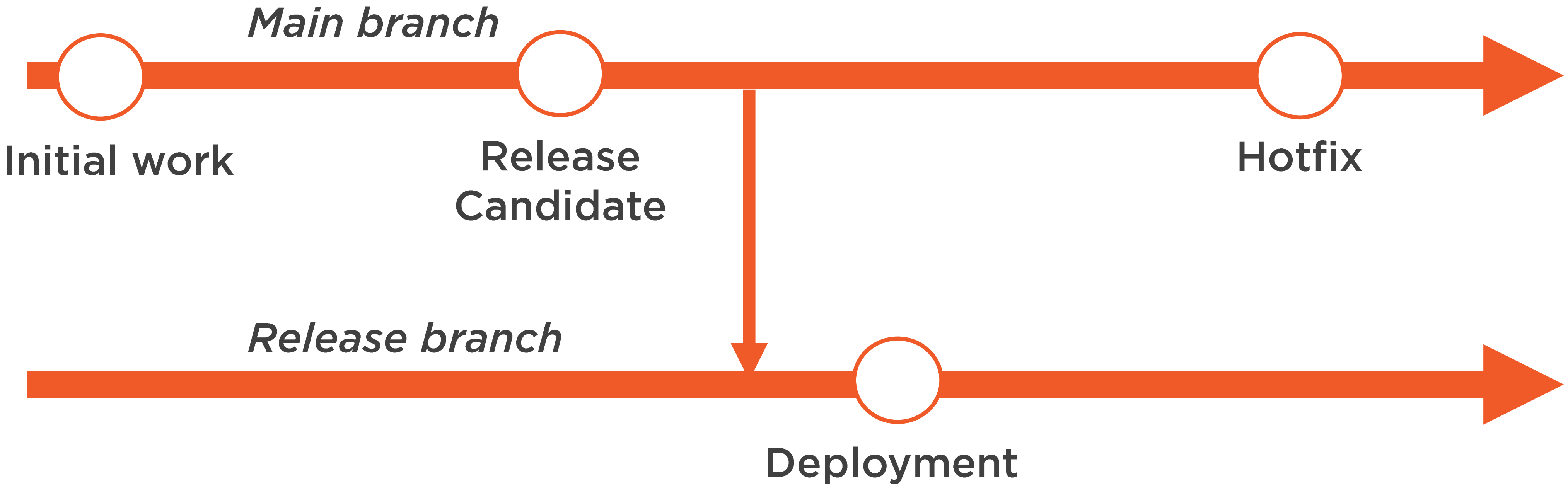
Deployment systems deploy – they don’t care *why* they’re deploying

Hotfixing is version control

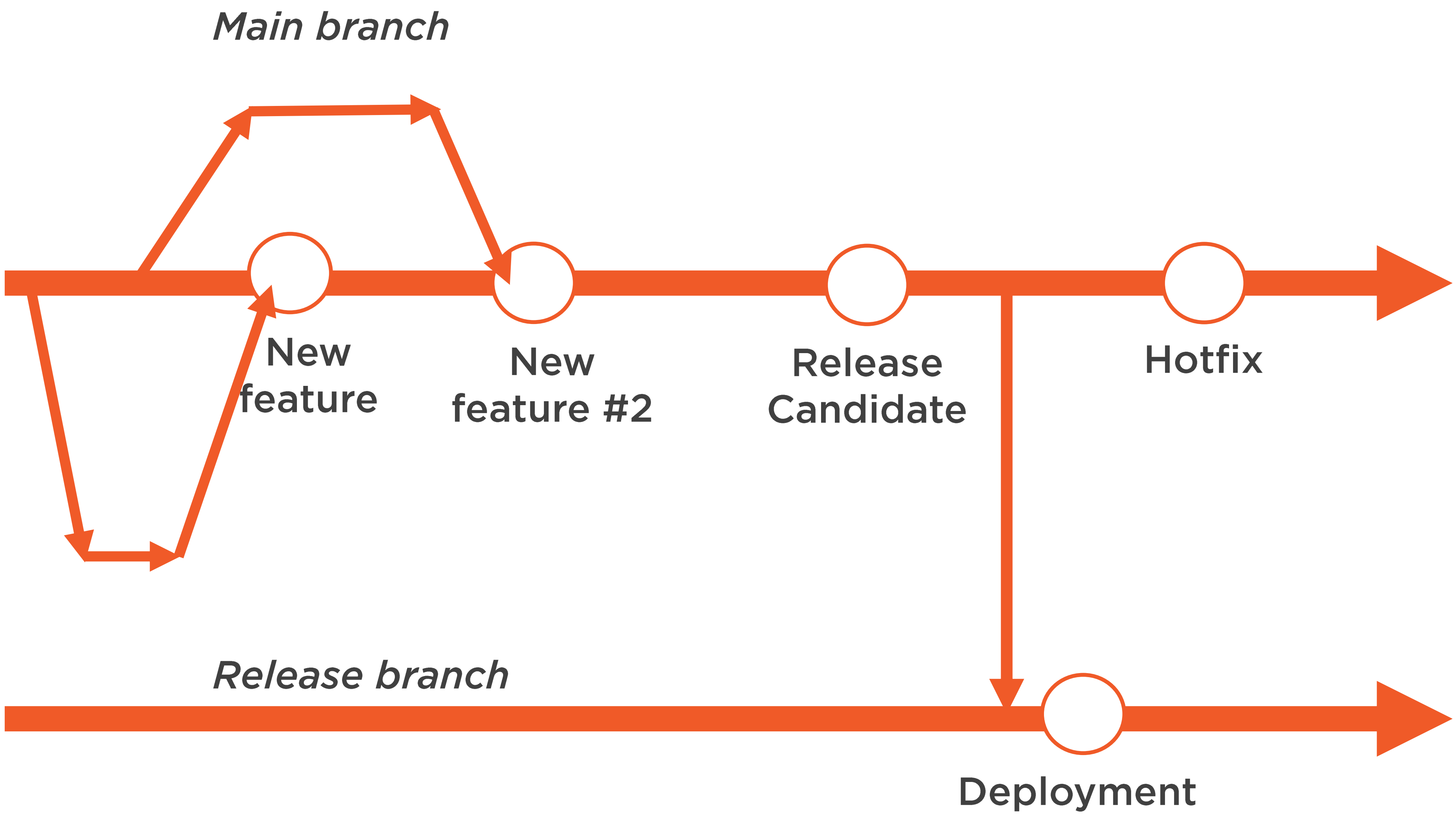
One Branch



A Release Branch



Many Short-lived Feature Branches



Where Do You Branch From?

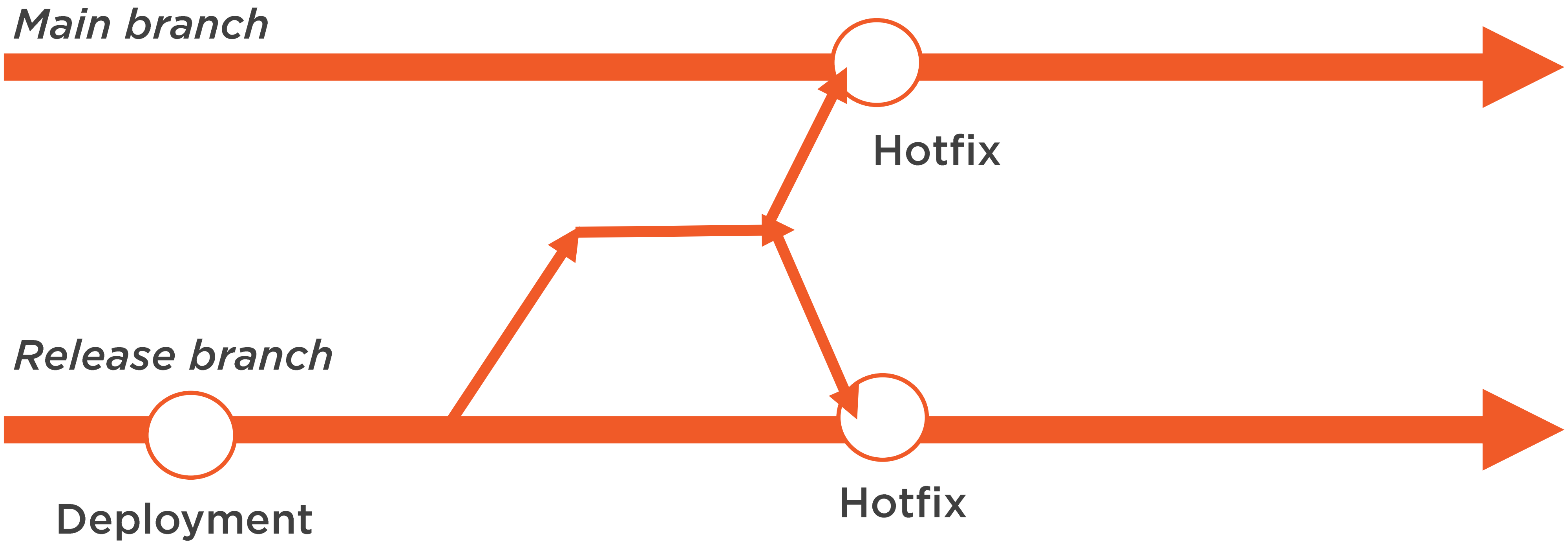
You've deployed a major fault to Production

Where do you hotfix from?

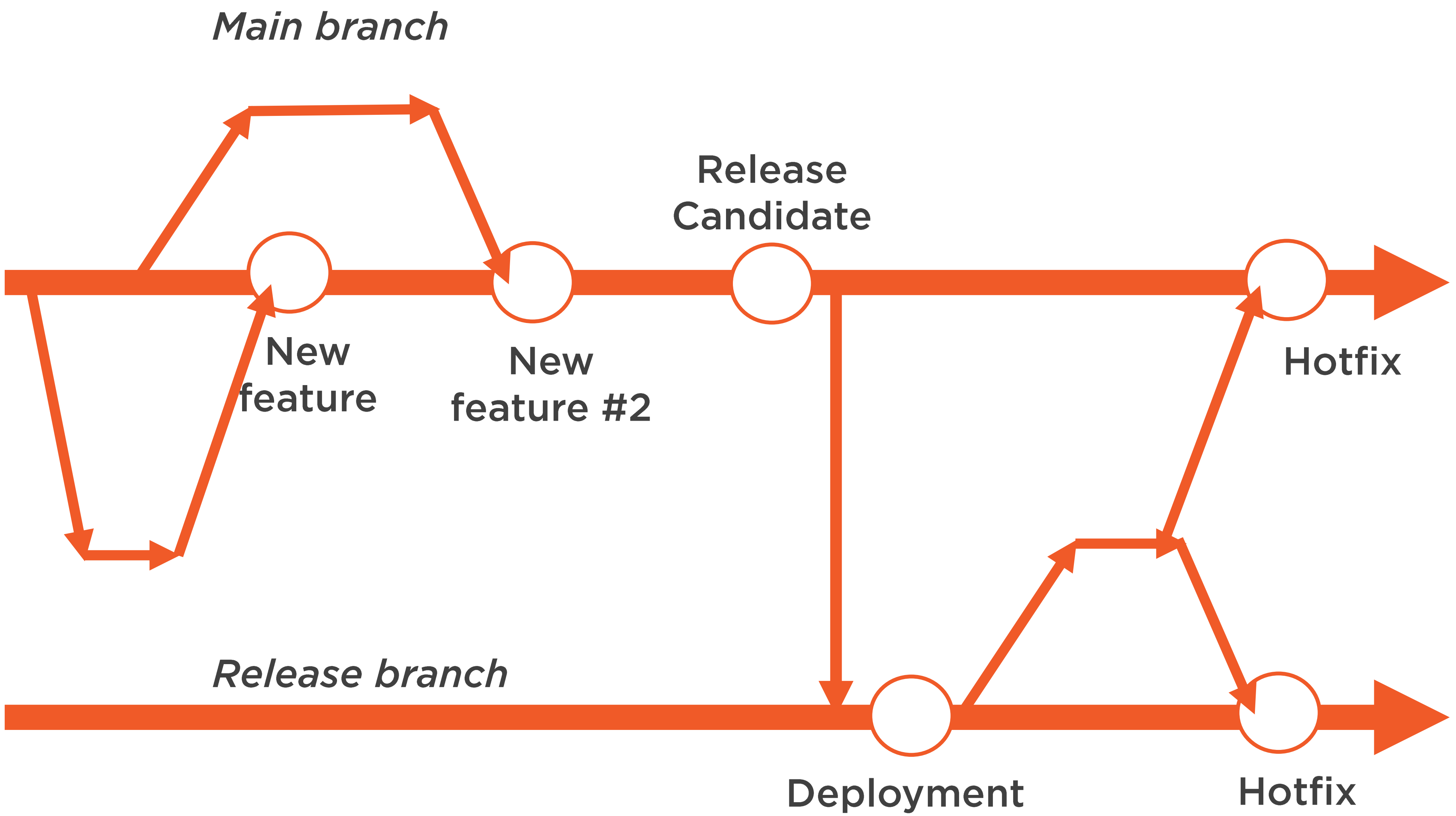
There's new stuff on the main branch

Hotfixes should be branched from the head of release branch

Many Short-lived Feature Branches



Moving Toward Gitflow



Nomenclature

Name your main branch whatever you feel like

Whatever you branch features off of is your main branch

If you periodically merge that branch to main for the purposes of release...

Then main is your release branch

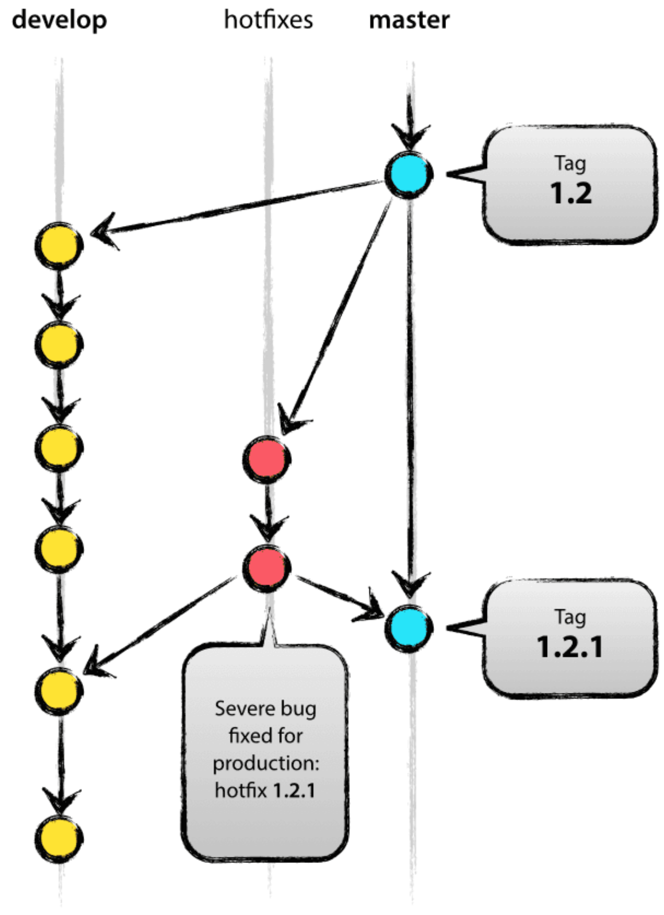
Each branch reflects a different (hopefully increasing) state of certainty about the suitability of the code for release

1. Feature – where we’re certain it’s not ready for production
2. Main or Develop – where we’re more certain, but we haven’t completed our verification process, and
3. Release – this reflects a commitment to ready for production.

GitFlow

1. Main
2. Develop
3. Feature
4. Release
5. Hotfix

GitFlow and Hotfixes



Vincent Driessen
https://bit.ly/37DZAIw

Hotfixes work pretty much the same

Merge back to main

And also, back to develop

Adopting a Git Workflow for Your Enterprise

Gitflow is not worth the trouble

ReleaseFlow

I don’t see the justification for the additional branch

GitFlow is controversial

Let your Git workflow emerge from use rather than being imposed top-down